**Plus de fonctionnalités, moins de tests ?**

Cameron travaillait comme développeur logiciel depuis environ huit mois dans une startup spécialisée dans les solutions fintech. L'entreprise développait un logiciel destiné à simplifier les approbations de prêts, en utilisant un ensemble de données plus large et l'apprentissage automatique pour accélérer les décisions.

Ayant commencé en tant que développeur junior, Cameron s'était rapidement fait remarquer et avait été promu à un poste de développeur intermédiaire après seulement six mois, grâce à son attitude proactive et à son talent pour résoudre des bugs complexes. Son ingénieur principal louait ses compétences en résolution de problèmes et notait qu'il avait une excellente compréhension du code et une solide maîtrise des principes logiciels.

Comme tous les développeurs de l'équipe, Cameron faisait partie d'une rotation d'astreinte, intervenant dès qu'un problème surgissait dans le système 24/7 de l'entreprise. Il était de garde une fois toutes les dix semaines, et bien que les incidents soient rares, l'équipe prenait les responsabilités d'astreinte très au sérieux, se préparant à toute éventualité.

Une nuit pendant son tour d'astreinte, Cameron reçut une notification urgente vers 23h30 : le système était en panne. Il réagit immédiatement. Après une brève investigation, il découvrit que le problème provenait d'un petit bug dans la nouvelle interface de demande de prêt qui venait d’être déployée. En explorant davantage les journaux, il remarqua autre chose : le système de notification, qui devait informer les demandeurs de l’état de leur prêt, échouait silencieusement depuis deux semaines.

Cameron corrigea rapidement le premier bug, poussa un correctif et mit à jour les canaux nécessaires sur Slack. Il décida de s'occuper du problème de notification le lendemain matin, pensant que ce n'était pas critique.

Le lendemain matin, Cameron se leva tôt, prit un café et commença à enquêter sur le problème de notification. En examinant le code, il réalisa que le service de notification n'avait pas de tests. Il écrivit une série de tests unitaires et, sans surprise, trois d'entre eux échouèrent. En creusant davantage, il découvrit qu'un déploiement récent avait introduit une erreur subtile qui bloquait l'envoi des notifications.

Lors du standup quotidien de l'équipe, Cameron présenta ses découvertes. Avant même qu'il puisse finir d'expliquer comment il résolvait le problème, son chef d'équipe et le responsable ingénierie se montrèrent visiblement inquiets. Ils l'interrompirent, lui demandant pourquoi il n'avait pas escaladé le problème immédiatement la nuit précédente. Ce que Cameron ignorait, c'est que l'entreprise avait une obligation légale d'informer les demandeurs de l'état de leur prêt sous 48 heures. Ne pas respecter ce délai pouvait entraîner des amendes de 10 000 $ par demandeur, pour chaque jour de retard.

Cameron sentit monter l'anxiété et la frustration, réalisant qu'il avait involontairement mis l'entreprise en danger.

En quelques minutes, une « salle de crise » fut formée, dirigée par le chef d'équipe, pour enquêter de toute urgence sur le problème. Pendant ce temps, le responsable ingénierie informait déjà les dirigeants de l'entreprise et l'équipe financière. Cameron expliqua à l'équipe qu'il avait écrit de nouveaux tests unitaires qui avaient identifié le problème, et ils travaillèrent ensemble pour corriger le code. En moins d'une heure, ils déployèrent le correctif en production.

Quelques semaines plus tard, lors de sa revue trimestrielle, Cameron reçut une évaluation décevante, qualifiée de « nécessite des améliorations ». Son manager cita la gestion du bug de notification comme raison principale.

Comment Cameron aurait-il pu gérer la situation autrement ?